12. Simboliai

Simbolių eilutė char[]

Simbolių sekas (simbolių eilutes) patogu laikyti char tipo masyvuose. Jie aprašomi panašiai, kaip ir kitų tipų masyvai, pavyzdžiui:

char s1[10];

Skirtumas tik tas, kad už paskutinės reikšmės masyve įrašomas eilutės pabaigos simbolis – nulis ('\0'). Simbolių eilutės char tipo dažnai vadinamos *C eilutėmis*.

Aprašant eilutę galima priskirti reikšmę, pavyzdžiui:

char s2[] = "C++ KALBA";

arba

char s2[12] = "C++ KALBA";

Pirmuoju atveju sukuriamas tokio ilgio masyvas s2, kuriame galėtų tilpti kabutėse užrašyta simbolių seka ir eilutės pabaigos simbolis, t. y. 10 elementų masyvas. Antruoju atveju kompiuterio atmintyje sukuriamas nurodyto dydžio masyvas. Jo pradžioje įrašoma kabutėse nurodyta simbolių seka:

| **Kintamasis\Indeksai** | **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** | **11** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Eilutės s2[] simboliai | C | + | + |  | K | A | L | B | A | \0 |  |  |
| Eilutės s2[12] simboliai | C | + | + |  | K | A | L | B | A | \0 |  |  |

Simbolių eilutėms apdoroti yra daug standartinių funkcijų. Jas sąlygiškai galima suskirstyti į tris grupes:

1) *Analizės funkcijos*– nekeisdamos eilutės randa kokių nors eilutės parametrų reikšmes.

2) *Konvertavimo ir kopijavimo funkcijos*– konvertuoja eilutę į skaičių arba kopijuoja ją į kitą eilutę.

3) *Modifikavimo funkcijos*– kokiu nors būdu keičia eilutės reikšmę.

Funkcijos, kurios suformuoja naują eilutę, išskyrus strncpy(), automatiškai įrašo jos pabaigos simbolį. Jei eilutė formuojama simbolis po simbolio, programuotojas pats turi įrašyti eilutės pabaigos simbolį. Naudojantis modifikavimo funkcijomis, būtina sekti, kad pakeistos eilutės ilgis nebūtų didesnis už nurodytą apraše.

Visos operacijos su char[] tipo eilutėmis (simbolių masyvais) atliekamos naudojant tam skirtas funkcijas.

*Eilučių analizės funkcijos*

| **Funkcijos prototipas** | **Paaiškinimai** |
| --- | --- |
| int **strcmp**(const char \*s1, const char \*s2); | Palygina eilutes s1 ir s2. Eilučių simbolių (ženklų) kodai iš eilės lyginami tol, kol randami skirtingi simboliai arba kurios nors eilutės pabaiga (didžiosios ir mažosios raidės skiriamos). Funkcija grąžina:   * 0, jei eilutės sutampa (s1 == s2); * teigiamąjį skaičių, jei eilutės s1 kodas didesnis už eilutės s2 kodą (s1 > s2); * neigiamąjį skaičių, jei eilutės s1 kodas mažesnis už eilutės s2 kodą (s1 < s2).   *Pavyzdys*  char e[] = "medis";  char s[] = "medelis";  if (strcmp(e, s) == 0) cout << "e = s";  else if (strcmp(e, s) < 0) cout << "e < s";  else cout << "e > s";  Rezultatas – pranešimas ekrane: e > s. |
| int **stricmp**(const char \*s1, const char \*s2); | Palygina eilutes s1 ir s2. Eilučių simbolių (ženklų) kodai iš eilės lyginami tol, kol randami skirtingi simboliai arba kurios nors eilutės pabaiga (didžiosios ir mažosios raidės neskiriamos). Funkcija grąžina:   * 0, jei eilutės sutampa (s1 == s2); * teigiamąjį skaičių, jei eilutės s1 kodas didesnis už eilutės s2 kodą (s1 > s2); * neigiamąjį skaičių, jei eilutės s1 kodas mažesnis už eilutės s2 kodą (s1 < s2).   *Pavyzdys*  char e[] = "medis";  char s[] = "Medis";  if (stricmp(e, s) == 0) cout << "e = s";  else if (stricmp(e, s) < 0) cout << "e < s";  else cout << "e > s";  Rezultatas – pranešimas ekrane: e = s. |
| int **strncmp**(const char \*s1, const char \*s2, int m); | Palygina eilučių s1 ir s2 pirmuosius m simbolių (didžiosios ir mažosios raidės skiriamos). Funkcija grąžina:   * 0, jei eilučių dalys (m pirmųjų simbolių) sutampa; * teigiamąjį skaičių, jei s1 dalis didesnė už s2 dalį; * neigiamąjį skaičių, jei s1 dalis mažesnė už s2 dalį.   *Pavyzdys*  char e[] = "medis"; char s[] = "medelis";  if (strncmp(e, s, 3) == 0) cout << "e = s";  else if (strncmp(e, s, 3) < 0) cout << "e < s";  else cout << "e > s";  Rezultatas – pranešimas ekrane: e = s. |
| int **strnicmp**(const char \*s1, const char \*s2, int m); | Palygina eilučių s1 ir s2 pirmuosius m simbolių (didžiosios ir mažosios raidės neskiriamos). Funkcija grąžina:   * 0, jei eilučių dalys (m pirmųjų simbolių) sutampa; * teigiamąjį skaičių, jei s1 dalis didesnė už s2 dalį; * neigiamąjį skaičių, jei s1 dalis mažesnė už s2 dalį.   *Pavyzdys*  char e[] = "medis";  char s[] = "Medelis";  if (strnicmp(e, s, 3) == 0) cout << "e = s";  else if (strnicmp(e, s, 3) < 0)cout << "e < s";  else cout << "e > s";  Rezultatas – pranešimas ekrane: e = s. |
| unsigned **strlen**(char \*s); | Apskaičiuoja ir grąžina eilutės s (be nulinio simbolio) ilgį.  *Pavyzdys*  char e[12] = "C++ kalba"; int n = strlen(e);  Rezultatas – kintamojo n reikšmė (9). |

*Eilučių konvertavimo ir kopijavimo funkcijos*

| **Funkcijos prototipas** | **Paaiškinimai** |
| --- | --- |
| double **atof**(const char \*s); | Konvertuoja eilutės s pirmuosius simbolius į realųjį skaičių, pateikiamą standartine išraiška, ir jį grąžina.  *Pavyzdys*  char \*s = "45.23e3abc";  double d = atof(s);  Rezultatas – kintamojo d reikšmė 45.230. |
| int **atoi**(const char \*s); | Konvertuoja eilutės s pirmuosius simbolius į sveikąjį skaičių ir jį grąžina.  *Pavyzdys*  char \*s = "45.23e3abc";  int n = atoi(s);  Rezultatas – kintamojo n reikšmė 45. |
| long **atol**(const char \*s); | Konvertuoja eilutės s pirmuosius simbolius į ilgąjį sveikąjį skaičių ir jį grąžina.  *Pavyzdys*  char \*s = "45.23e3abc";  long n = atol(s);  Rezultatas – kintamojo n reikšmė 45. |
| char \***strcpy**(char \*d,  const char \*s); | Kopijuoja eilutę s į eilutę d ir grąžina rodyklę į eilutę d.  *Pavyzdys*  char e[12];  strcpy(e, "C++ kalba");  Rezultatas – eilutės e reikšmė: "C++ kalba". |
| char \***strncpy**(char \*d, char \*s,  unsigned m); | Kopijuoja m simbolių iš eilutės s į eilutę d ir grąžina rodyklę į eilutę d (eilutės pabaigos simbolio ('\0') neįrašo).  *Pavyzdys*  char e[12]; char s[] ="C++ kalba";  strncpy(e, s, 3); e[3]='\0';  Rezultatas – eilutės e reikšmė: "C++". |
| char \***strlwr**(const char \*s); | Perrašo visą eilutę s mažosiomis raidėmis ir grąžina rodyklę į pakeistą eilutę s (mažųjų raidžių ir kitų simbolių nekeičia).  *Pavyzdys*  char e[] = "C++ Builder";  strlwr(e);  Rezultatas – eilutės e nauja reikšmė: "c++ builder". |
| char \***strupr**(const char \*s); | Perrašo visą eilutę s didžiosiomis raidėmis ir grąžina rodyklę į pakeistą eilutę s (didžiųjų raidžių ir kitų simbolių nekeičia).  *Pavyzdys*  char e[] = "C++ Builder";  strupr(e);  Rezultatas – eilutės e nauja reikšmė: "C++ BUILDER". |

*Eilučių modifikavimo funkcijos*

| **Funkcijos prototipas** | **Paaiškinimai** |
| --- | --- |
| char \***strcat**(char \*d,  const char \*s); | Prie eilutės d prijungia eilutę s ir grąžina rodyklę į sujungtą eilutę d.  *Pavyzdys*  char e[12] = "C++ "; char s[] = "kalba";  strcat(e, s);  Rezultatas – eilutės e nauja reikšmė: "C++ kalba". |
| char \***strncat**(char \*d,  const char \*s, unsigned m); | Prie eilutės d prijungia m simbolių iš eilutės s ir grąžina rodyklę į sujungtą eilutę d.  *Pavyzdys*  char e[15] = "C++ ";  char s[] = "programavimas";  strncat(e, s, 8);  Rezultatas – eilutės e nauja reikšmė: "C++ programa". |
| char \***strnset**(char \*s, char c,  unsigned m); | Užpildo s eilutės m elementų simboliu c ir grąžina rodyklę į eilutę s.  *Pavyzdys*  char e[] = "C++ Builder";  strnset(e, '\*', 4);  Rezultatas – eilutės e nauja reikšmė: "\*\*\*\*Builder". |
| char \***strset**(char \*s, char c); | Užpildo eilutę s simboliu c ir grąžina rodyklę į eilutę s.  *Pavyzdys*  char e[] = "C++ Builder";  strset(e, '\*');  Rezultatas – eilutės e nauja reikšmė: "\*\*\*\*\*\*\*\*\*\*\*". |

Eilučių skaitymas ir rašymas

Simbolių eilutėms įvesti naudojamas operatorius >>. Jis priskiria įvesties sraute nurodytam simbolių eilutės kintamajam visus simbolius iki pirmo tarpo simbolio arba iki eilutės pabaigos simbolio '\n' (visi tarpo simboliai iki eilutės pradžios praleidžiami). Eilutės pabaigoje įterpia eilutės pabaigos simbolį '\0'. Jeigu aprašant eilutę pabaigos simbolio vieta nebuvo numatyta arba eilutė įvesties sraute yra ilgesnė, nei numatyta eilutės apraše, tuomet eilutės pabaigos simbolis į ją neįrašomas ir veiksmai su tokia eilute tampa klaidingi arba neįmanomi.

Pavyzdžiui, jei įvesties sraute fd yra simbolių eilutė „Man patinka C++ programavimo kalba“ tai, atlikę veiksmus

char eil[100];

fd >> eil; // fd – įvesties srautas

cout << eil;

ekrane matysime simbolių eilutę:

*Man*

Norint įvesti simbolių eilutę ne iki tarpo, reikia naudoti globaliąją funkciją get():

istream & get(char eil[], int n);

Funkcija get() iš įvesties srauto įveda n-1 simbolį ir palieka vietą eilutės pabaigos simboliui '\0'. Pavyzdžiui, jeigu įvesties sraute fd yra simbolių eilutė „Man patinka C++ programavimo kalba“ tai, atlikę veiksmus

char eil[100];

fd.get(eil, 10);

cout << eil;

ekrane matysime tokią simbolių eilutę:

*Man patin*

Įvesties sraute iki eilutės pradžios esantys tarpo simboliai nepraleidžiami. Jeigu įvesties sraute yra mažiau simbolių, nei numatyta eilutės apraše, tuomet įvestis baigiama, aptikus eilutės pabaigos simbolį. Jeigu reikia įvesti tiek simbolių, kiek nurodyta eilutės apraše, tuomet patogu naudoti operatorių sizeof, kuris apskaičiuoja kintamojo ilgį baitais:

fd.get(eil, sizeof eil);

***Pavyzdys.*Funkcija iš srauto fd nukopijuoja n eilučių į srautą fr.**

void KopijuotiSrautoEilutes(ifstream & fd, ofstream & fr, int n)

{

char E[100];

for (int i = 0; (i < n) && (!fd.eof()); i++) {

fd.get(E, sizeof E);

fr << E << endl;

}

}

Simbolių eilutėms išvesti naudojamas operatorius <<. Jis išveda visus eilutės simbolius iki jos pabaigos simbolio '\0', pavyzdžiui:

fr << eil;

Simbolių masyvas

Veiksmams su simbolių eilutėmis atlikti yra daug funkcijų. Tačiau simbolių eilutės – tokie pat masyvai, kaip sveikųjų ar realiųjų skaičių. Tik simbolių masyvo elementų reikšmės yra simboliai. Todėl darbui su simbolių masyvo elementais galima taikyti tuos pačius algoritmus, kaip ir su sveikųjų ar realiųjų skaičių masyvų elementais.

***Pavyzdys.*Simboliai skaitomi iš failo.**

void Skaityti(char E[], int & n)

{

ifstream fd("Sim.txt");

for (n = 0; !fd.eof() && n < Cn; n++) // Cn – masyvo dydis

fd >> E[n];

fd.close();

}

Operatorius >> praleidžia visus tarpo, tabuliavimo žymės ir eilutės pabaigos simbolius. Norint masyve laikyti visus tekste esančius simbolius, įvedimo sakinį reikia pakeisti tokiu:

fd.get(E[n]);

***Pavyzdys.*Rašomos simbolių masyvo reikšmės.**

void Spausdinti(char E[], int n)

{

ofstream fr("SimRez.txt");

for (int i = 0; i < n; i++)

fr << E[i];

fr.close();

}

***Pavyzdys.*Skaičiuojama, kiek kartų masyve pasitaiko nurodytas simbolis.**

int Kiek(char E[], int n, char sim)

{

int k = 0;

for (int i = 0; i < n; ++)

if (E[i] == sim) k++;

return k;

}

Norint išsiaiškinti, kiek a raidžių yra masyve, reikia parašyti kreipinį į funkciją Kiek():

cout << Kiek(E, n, 'a');

Pavyzdžiui, norint rasti, kuri mažoji raidė simbolių masyve pasikartoja dažniausiai, taip pat galima pasinaudoti jau turima Kiek() funkcija:

char Kuri(char E[], int n)

{

int sk, k = 0;

char sim = ' ';

for (char i = 'a'; i <= 'z'; i++) {

sk = Kiek(E, n, i);

if (sk > k) {

k = sk;

sim = i;

}

}

return sim;

}

Jeigu simbolių masyvo gale nėra eilutės pabaigos simbolio '\0', tuomet negalima taikyti darbo su simbolių eilutėmis funkcijų ir kitų veiksmų visam masyvui, pavyzdžiui, cout << E;

Simbolių eilučių masyvas

Viena simbolių eilutė yra simbolių masyvas. Jeigu eilučių yra daug, patogu jas susirašyti į vieną masyvą, pavyzdžiui:

char Mas[100][15];

Šiame masyve gali būti laikoma 100 eilučių, kurių kiekviena gali turėti po 15 simbolių. Pavyzdžiui,

cout << Mas[5] << endl; // parodo ekrane penktoje vietoje esančią eilutę

cout << Mas[2][12] << endl; // parodo ekrane antroje vietoje esančios eilutės 12-ą simbolį

***Pavyzdys.*Tekstas iš failo į skaitomas į simbolių eilučių masyvą.**

void Skaityti(char Mas[][15], int & n)

{

ifstream fd("Simb.txt");

for (n = 0; !fd.eof() && n < CMax; n++) {

fd.get(Mas[n], sizeof Mas[n]);

fd.ignore(80, '\n');

// praleidžiami visi likę simboliai iki eilutės pabaigos

// paprastai tekstiniuose failuose nebūna ilgesnių kaip 80 simbolių eilučių

}

fd.close();

}

Masyvo eilutes išvesties sraute fr galima spausdinti taip:

for (int i = 0; i < r; i++)

fr << A[i] << endl;

Užrašius fr << A[3][5] << endl; bus išspausdintas masyvo trečioje vietoje esančios eilutės penktas simbolis.